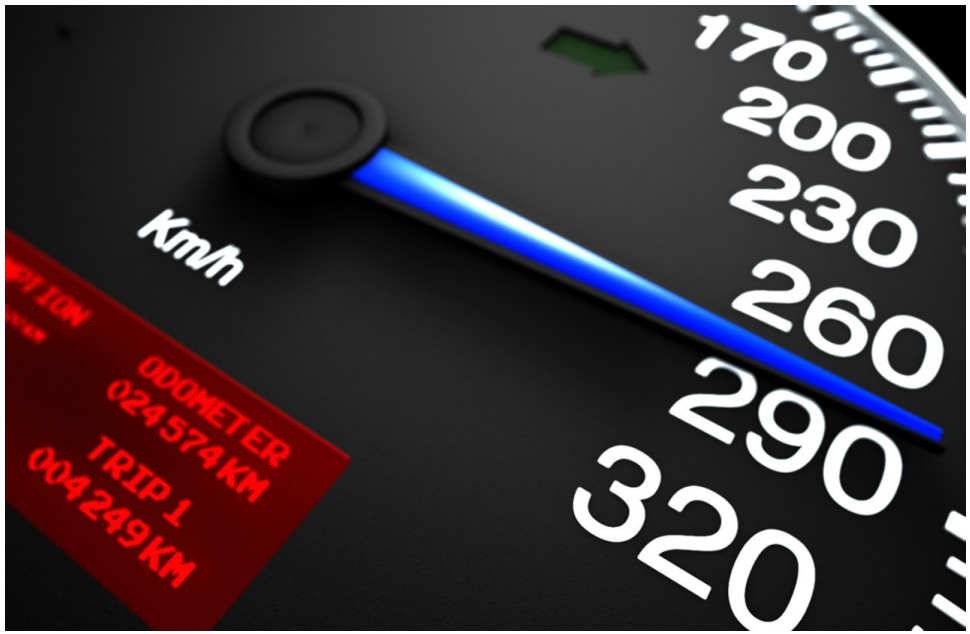
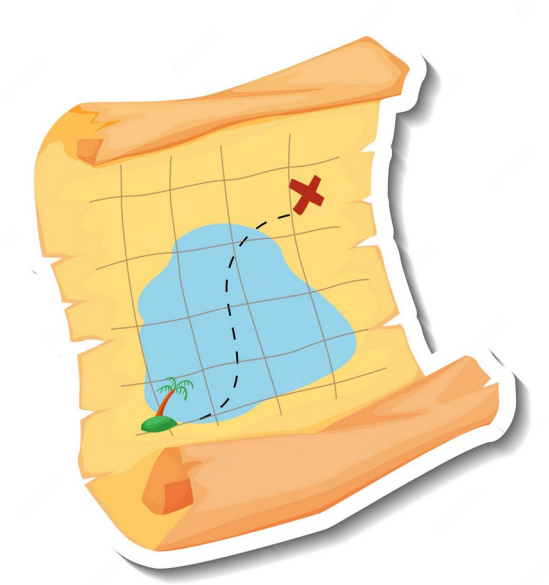


Analisi prestazionale di classificatori sulla difettosità del codice a seguito dell'applicazione o meno di metodi di Feature Selection

Daniele La Prova - 0320429

Outline

- Introduzione
- Metodologia
- Discussione Risultati
- Conclusioni



Introduzione

- Si vuole esaminare l’andamento dei parametri prestazionali di tre classificatori al variare dell’applicazione o meno di tecniche di Feature Selection, addestrati e valutati su un dataset rappresentante la difettosità delle risorse dei due progetti considerati
- I classificatori in esame sono NaiveBayes, RandomForest, IBK
- Le tecniche di FS considerat dono Filter e Wrapper (euristiche greedy). Per entrambi, si considerano Backward e Forward search
- I Progetti considerati sono Apache/BookKeeper e Apache/Avro
- Si esamineranno anche la rilevanza ai fini del modello di alcune features dei dataset presi in considerazione

Metodologia

- Per poter trarre delle conclusioni è necessario prima raccogliere dei dati da analizzare
- È stato costruito un dataset contenente valori di metriche prescelte per ogni file dei progetti considerati. Tali metriche sono state scelte poiché si sospetta che siano correlate con la presenza di bug.

Metodologia – Metriche

- PUBLICS: numero di attributi e metodi pubblici
  - Un alto numero indica che buona parte della classe è esposta al resto del sistema, e dunque prona a cambiamenti
- CODE_SMELLS: numero di smells
  - Un alto numero implica un debito tecnico elevato, e dunque una difficile mantenibilità
- AGE: numero di settimane che intercorrono dalla creazione della risorsa al momento della misurazione
  - Può essere interessante da considerare insieme ad altre metriche
- N_AUTH: Numero di autori diversi dei commit che riguardano una risorsa.
  - Ogni autore deve comprendere appieno le modifiche degli altri per poter effettuare modifiche senza introdurre errori
- SIZE: LOCs della risorsa
  - Un numero alto offre più superficie per l'introduzione di bugs

Metodologia – Metriche

- N_R: Numero di commits che riguardano la risorsa dalla sua creazione fino al momento della misurazione
  - Un valore alto rispetto a AGE indica che la risorsa ha subito numerose modifiche in un lasso breve di tempo, e dunque è stata ad alto rischio di introduzione bug
- N_FIX: Numero di commit che riguardano la risorsa a cui corrisponde un ticket su JIRA identificato come bug.
  - Un alto valore suggerisce che la risorsa è prona all'introduzione di bug, o che comunque è difficile da aggiustare
- CHURN: added – deleted LOCs della risorsa nel periodo che intercorre tra la release considerata per la misurazione e quella precedente
  - Un alto valore in modulo indica che la risorsa ha subito numerose modifiche, ovvero numerose occasioni per introdurre dei bug
- DEPENDENCIES: Numero di `import` + `extends` + `implements`
  - Un alto numero di dipendenze può implicare un alto rischio di regressioni

Metodologia – Misurazioni

- Le metriche del software dei progetti sono raccolte usando il programma `buggynessCrawler`
- È possibile specificare in input il progetto da considerare, le metriche da misurare e quale percentuale di ultime releases vanno scartate per mitigare l'effetto di Snoring
- Per ogni release di ogni progetto si effettua il checkout del codice, e per ogni risorsa di esso si eseguono istanze di `Miner`, ognuna delle quali implementa una misura per una metrica e la applica sulla risorsa considerata
- Tutti i risultati delle misurazioni sono registrati in una `ObservationMatrix` dove sono identificati dal nome della release in misurazione e da quello della risorsa
  - ESEMPIO: nella cella `i`, `j` è registrato un risultato per ogni metrica considerata che riguarda la release `i` e la risorsa `j`
- La matrice così prodotta è infine stampata in un file CSV

Metodologia – Buggyness

- Anche la BUGGYNESS è considerata come una metrica ai fini dell'implementazione, dunque esiste nel programma un `Miner` dedicato ad effettuarne la misurazione
- Per poter scoprire quali classi sono buggy, si applica il seguente algoritmo:
  - Si raccolgono tutti i ticket su JIRA riferiti al progetto in esame che riguardano un bug, in ordine cronologico dal più vecchio al più recente
  - Per ogni ticket, si prende la piu recente FV registrata e si ricerca nel log della repository il commit più recente che cita l'id del ticket nel commento e precede la suddetta FV (il *fixCommit*)
  - Per ogni file `i` citato nel changeset del commit, si registra il valore `yes` per la `Feature` BUGGYNESS per ogni release `j` che intercorre tra la IV inclusa e la FV esclusa
- Si scartano tutti i tickets che presentano almeno una delle seguenti caratteristiche:
  - FV = OV (*non-production* bug);
  - ∄ *fixCommit* associato;
  - ∄ neanche una FV associata consistente;
  - FV < IV (per costruzione dell'algoritmo).

Metodologia – Proportion

- Per ogni bug, se si conosce la IV si aggiorna la stima del valore di Proportion (P) calcolata come *Exponential Weighted Moving Average* (EWMA)
  - Tale valore è inizializzato a 1, implicando che IV = OV. Tale assunzione è sembrata più ragionevole in termini di correttezza e sforzo rispetto a ricorrere a Cold Start oppure inizializzare P = 0 (=> IV = FV).
  - L'uso di EWMA implica che nel calcolo della media il peso dei campioni tende a zero tanto più sono vecchi, unendo i vantaggi della facilità di calcolo dell'approccio Incremental e tenendo presente l'influenza del tempo e la granularità a livello di ticket del Moving Window, al costo di un'inaccuratezza del calcolo tanto più accentuata quanto il set di campioni è ridotto
- Se non la si conosce, si calcola la IV usando la stima di P corrente.
  - Valore di P è calcolato solo su tickets passati, elimina il bisogno di relabeling quando si effettueranno splits del dataset

Metodologia – Evaluation

- I files csv prodotti da `buggynessCrawler` sono dati in input al programma `classifier-evaluator`, insieme a una lista di nomi di classificatori, una lista di metodi di *Feature Selection* (FS), e il senso di ricerca delle features (ignorato se non si vuole applicare FS)
- Per ogni dataset in input, il programma:
  - Applica i metodi di FS desiderati;
  - Valuta i classificatori desiderati con il dataset corrente;
  - Registra i risultati in una lista di oggetti `RichEvaluation`
  - Stampa i risultati in un file csv.
- I classificatori sono valutati tramite l'algoritmo *WalkForward* (WF), implementato come un generatore di oggetti `RichEvaluation`
  - Tale classe estende `Evaluation` , offerta da Weka, per poter registrare dati aggiuntivi come il numero di iterazioni, …
- Ad ogni iterazione, il generatore effettua lo split del dataset in memoria e aggiorna train e test set secondo WF, valuta il classificatore e genera la valutazione contenente i dati delle prestazioni

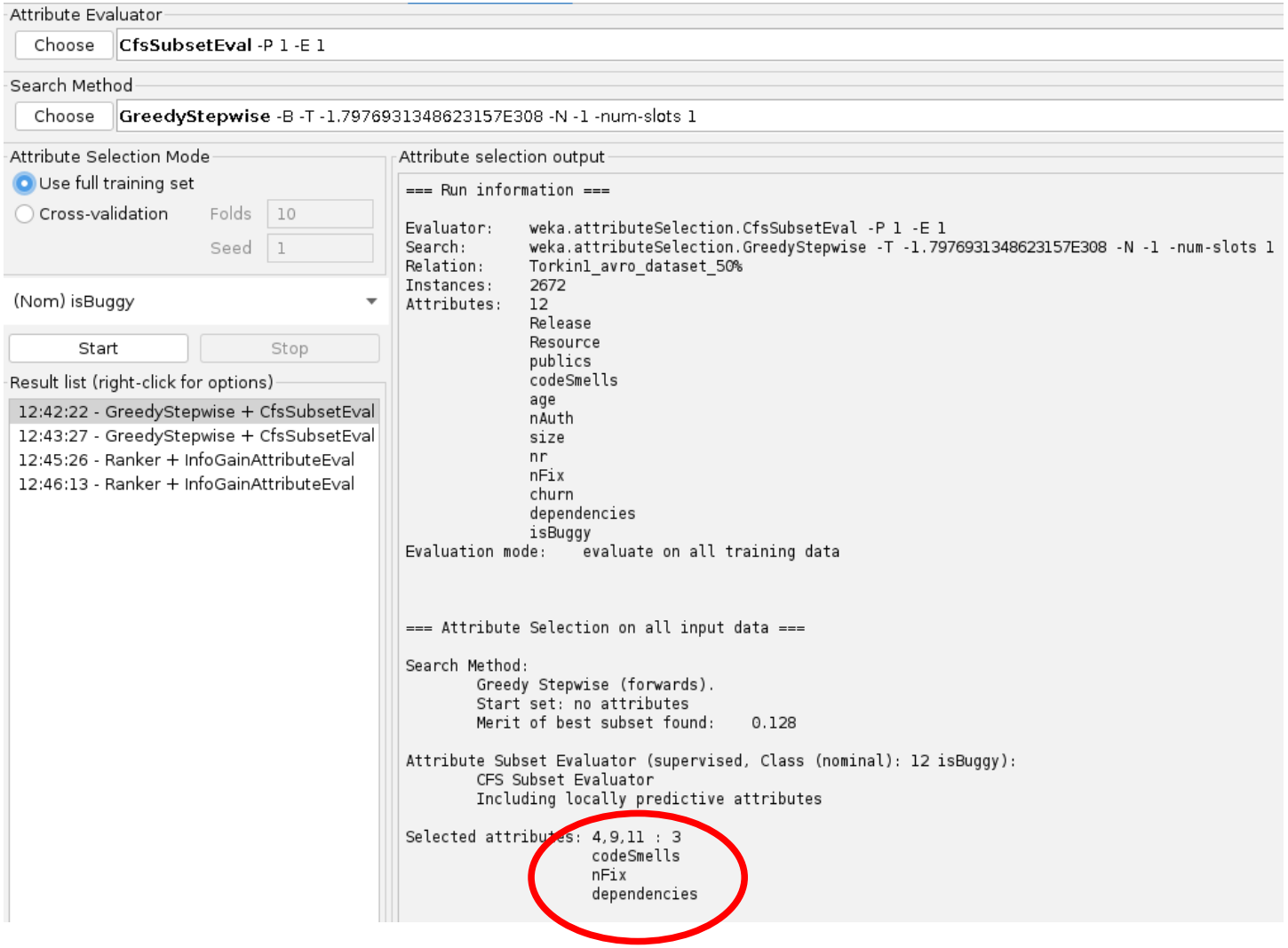
Discussione Risultati

- In questa sezione sono analizzati i dati ottenuti dall'esecuzione di `classifier-evaluator` valutando combinazioni di approcci di Features e selection.
- I dataset csv ottenuti sono stati analizzati e graficati con i seguenti programmi open source:
  - *SOFA – Statistics Open For All* 
  - *Weka 3 GUI: Machine Learning Software in Java* 
- È stato scartato la metà più recente delle releases per mitigare l'effetto di Snoring
- La threshold è impostata a 0.5 (predizioni hanno lo stesso costo)

```
1   "compounds": [
2     {
3       "name": "Fire all systems",
4       "configurations": [
5         "evaluation only",
6         "FS (forward wrapper)",
7         "FS (backward wrapper)",
8         "FS (forward filter)",
9         //"FS (backward filter)",
10         "FS (forward filter, wrapper)",
11         //"FS (backward filter, wrapper)"
12       ],
13       "stopAll": true
14     }
15  ]
16
```

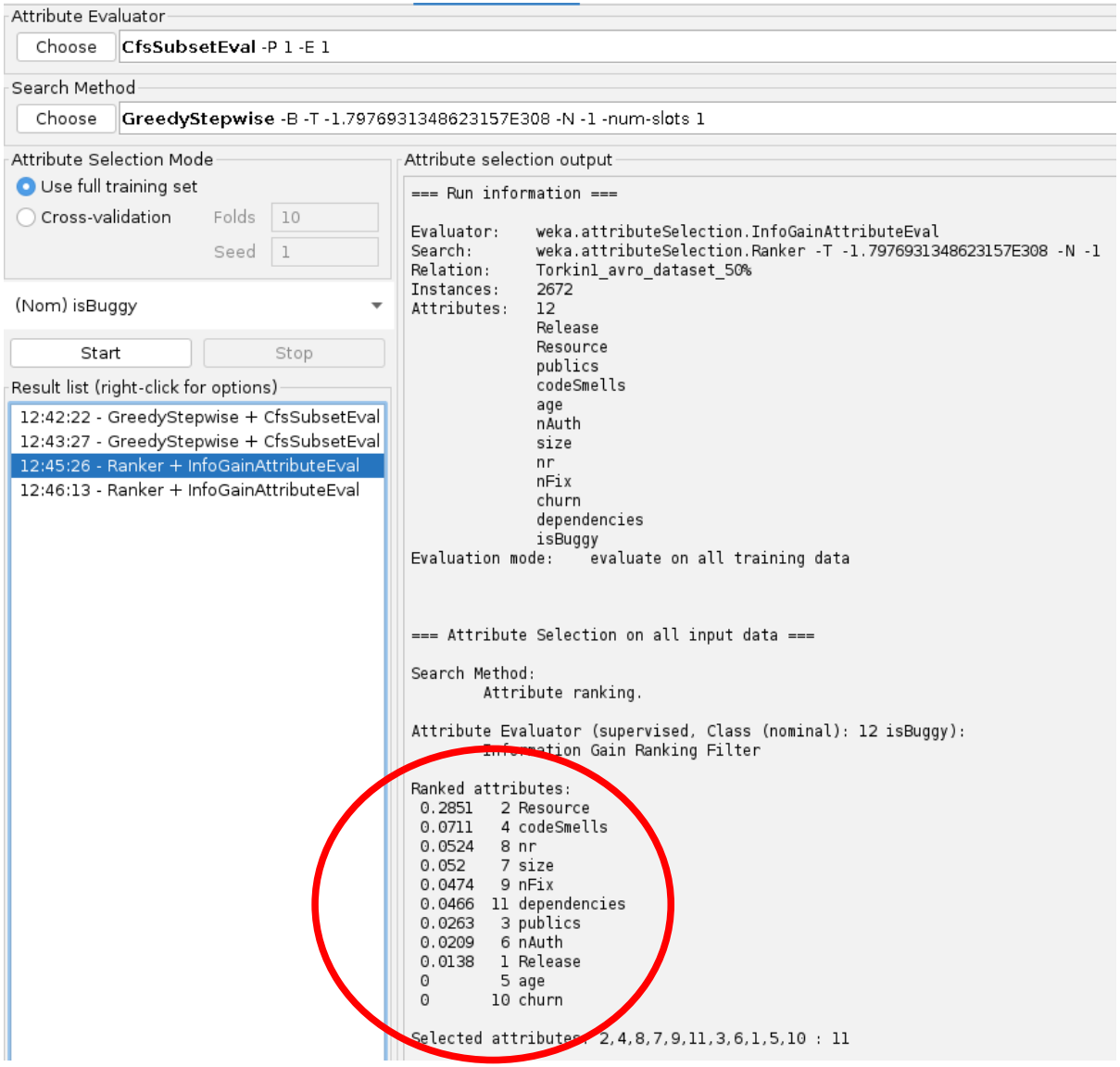
- Le combinazioni che comprendono Wrapper + Backward Search non sono state implementate al momento
- È necessario implementare un modo per impedire a FS di scartare l'attributo "Release" del dataset
- Forse in lavori futuri ...

Discussione Risultati – Features



- Nella sezione Metodologia – Metriche sono state presentate le metriche PUBLICS e DEPENDENCIES non presenti nel pool di scelta, ma che sono state ritenute dall'autore rilevanti per la previsione di bug
- Per verificare se tali metriche fossero rilevanti per il modello è stata eseguita una FS per vedere se esse avrebbero fatto parte del sottoinsieme risultante
- Cercando euristicamente il **sottoinsieme di metriche** più efficace, è risultato farne parte solo DEPENDENCIES tra le due sopracitate, insieme a CODE_SMELLS e N_FIX

Discussione Risultati – Features



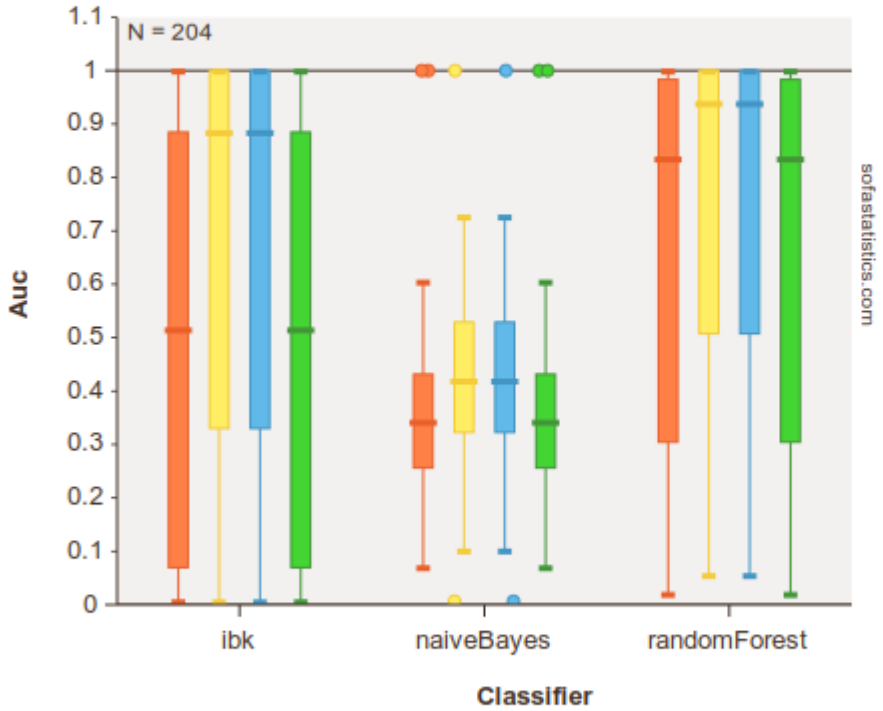
- Cercando la metrica che **da sola** risulta più efficace per il modello, se si escludono gli attributi Resource e Release, è risultato che DEPENDENCIES e PUBLICS sono rispettivamente la 5^ e la 6^ metrica piu efficace
- Inoltre, PUBLICS sembra essere in netto distacco come punteggio rispetto alle posizioni superiori, mentre DEPENDENCIES si distacca poco dalla posizione immediatamente sopra
- La metrica CODE_SMELLS sembra essere determinante in entrambi i casi

Discussione Risultati - AUC

From sofa_db.all on 14/06/2022 at 11:08

Data filtered by: `dataset_name` = 'Torkin1_avro_dataset_50%' AND `Is_Feature_Search_Backward?` = 'false'

AUC by Classifer, series by FS approach
(Precison - Recall curve)



Feature_Selection_Approaches: [filter wrapper] [filter] [none] [wrapper]

Outliers displayed. Lower whiskers are 1.5 times the Inter-Quartile Range below the lower quartile, or the minimum value, whichever is closest to the middle. Upper whiskers are calculated using the same approach.

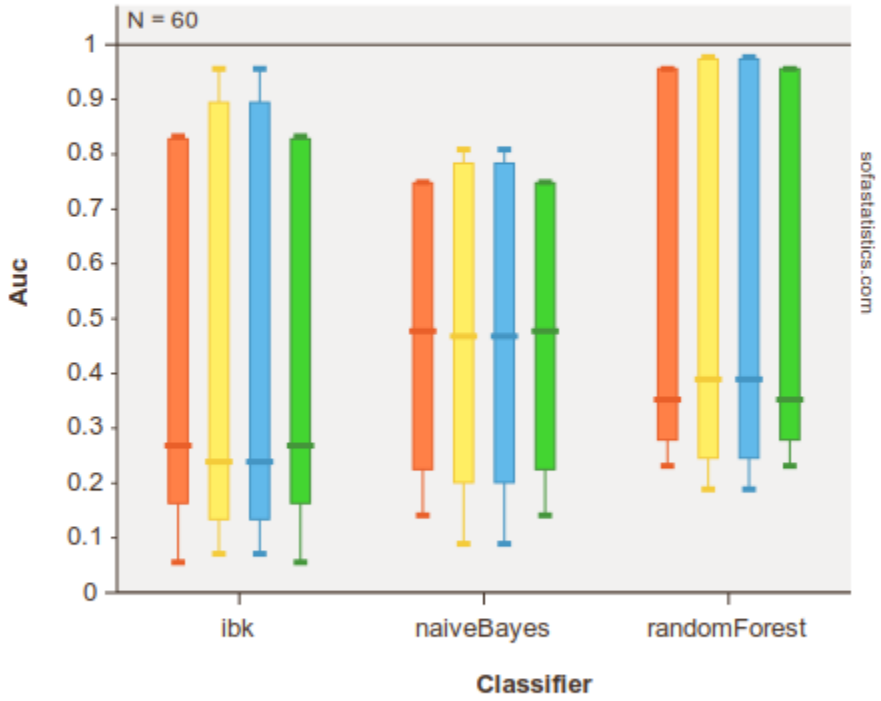
- Considerando i dati del progetto Avro, L'approccio Filter non sembra apportare miglioramenti
- L'approccio Wrapper sembra peggiorare le prestazioni
- Sebbene possa sembrare un risultato inaspettato, una spiegazione può provenire dalla ridotta dimensione dell'insieme delle metriche considerate (9). Probabilmente, esse sono in un numero molto inferiore rispetto a quello ottimale, e per ottenere prestazioni migliori bisognerebbe dunque aggiungere attributi anziché sottrarli
- L'esito non cambia se si usa la backward search invece che la forward

Discussione Risultati - AUC

From sofa_db.all on 15/06/2022 at 06:11

Data filtered by: `Dataset_Name`='Torkin1_bookkeeper_50%_dataset' AND (`Is_Feature_Search_Backward?`='false' OR `Feature_Selection_Approaches`='[none]')

AUC by Classifier, series by FS approach
Computed on Precision by Recall curve



Feature_Selection_Approaches: [filter wrapper] [filter] [none] [wrapper]

Outliers displayed. Lower whiskers are 1.5 times the Inter-Quartile Range below the lower quartile, or the minimum value, whichever is closest to the middle. Upper whiskers are calculated using the same approach.

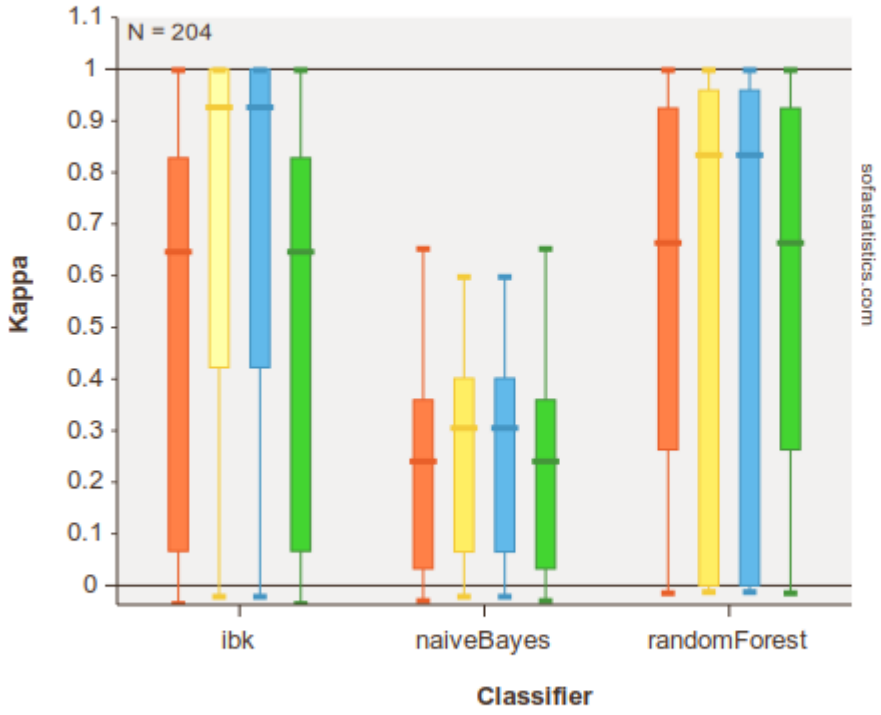
- Considerando il progetto BookKeeper, si hanno dei leggeri miglioramenti nel valore medio di AUC con Ibk e naiveBayes
- Inoltre, tali classificatori assumono valori per AUC in un range più ampio quando è applicato l'approccio Filter o nessun approccio
- Per randomForest si applicano le stesse considerazioni viste con Avro

Discussione Risultati - Kappa

From sofa_db.all on 15/06/2022 at 04:49

Data filtered by: `Dataset_Name`='Torkin1_avro_dataset_50%' AND `Is_Feature_Search_Backward?`='false'

Kappa by Classifier, series by FS approach



Feature_Selection_Approaches: [filter wrapper] [filter] [none] [wrapper]

Outliers displayed. Lower whiskers are 1.5 times the Inter-Quartile Range below the lower quartile, or the minimum value, whichever is closest to the middle. Upper whiskers are calculated using the same approach.

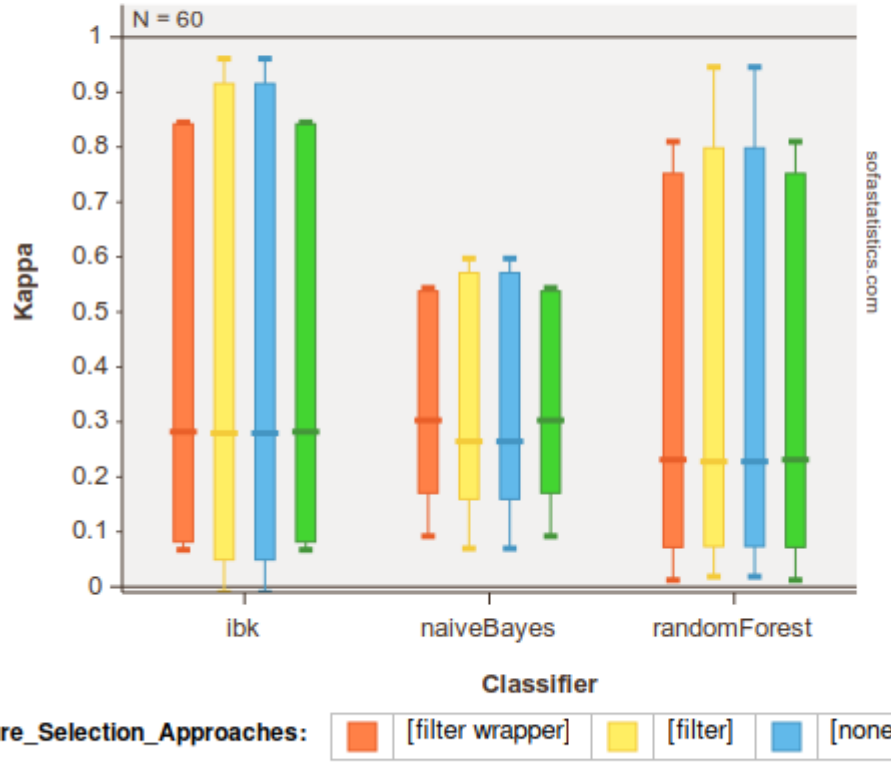
- Con Avro, si può notare come ibK abbia mostrato delle prestazioni peggiori rispetto a un classificatore dummy a seguito dell'applicazione di FS wrapper
- NaiveBayes peggiora leggermente
- RandomForest assume valori in un range più elevato a seguito di FS wrapper, tuttavia la media dei valori è inferiore rispetto a Filter o nessuna FS

Discussione Risultati - Kappa

From sofa_db.all on 15/06/2022 at 05:48

Data filtered by: `Dataset_Name`='Torkin1_bookkeeper_50%_dataset' AND (`Is_Feature_Search_Backward?`='false' OR `Feature_Selection_Approaches`='[none]')

Kappa by Classifier, series by FS approach



Outliers displayed. Lower whiskers are 1.5 times the Inter-Quartile Range below the lower quartile, or the minimum value, whichever is closest to the middle. Upper whiskers are calculated using the same approach.

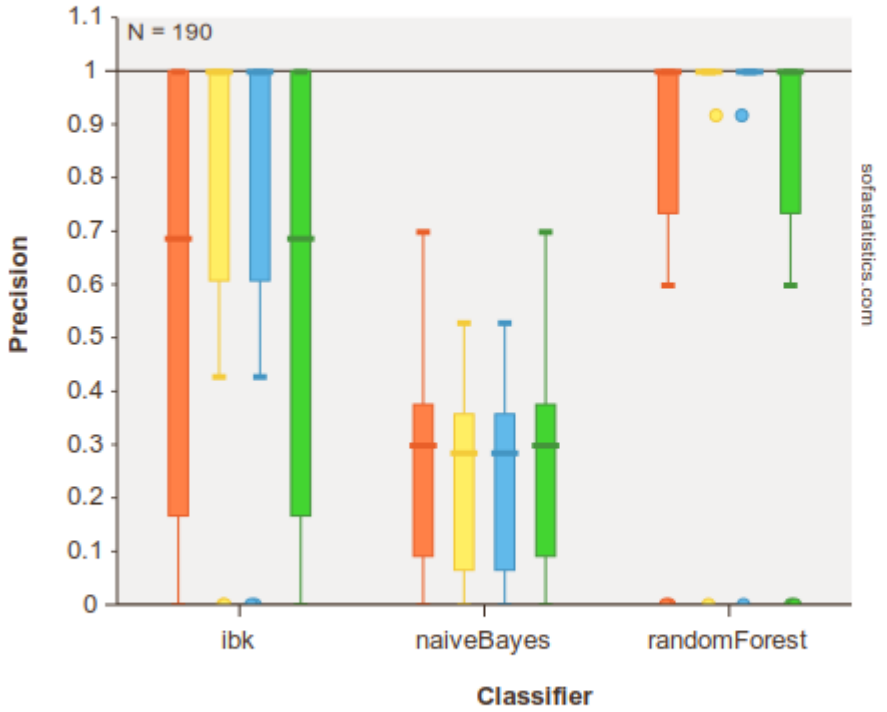
- Con BookKeeper, ibK e randomForest non subiscono notevoli differenze, se si esclude un range di valori assunti più alto quando è applicato Wrapper
- NaiveBayes in media dimostra prestazioni migliori rispetto a classificatore dummy quando è applicato Wrapper FS

Discussione Risultati - Precision

From sofa_db.all on 15/06/2022 at 04:54

Data filtered by: `Dataset_Name`='Torkin1_avro_dataset_50%' AND `Is_Feature_Search_Backward?`='false'

Precision by Classifier, series by FS approach



Feature_Selection_Approaches: [filter wrapper] [filter] [none] [wrapper]

Outliers displayed. Lower whiskers are 1.5 times the Inter-Quartile Range below the lower quartile, or the minimum value, whichever is closest to the middle. Upper whiskers are calculated using the same approach.

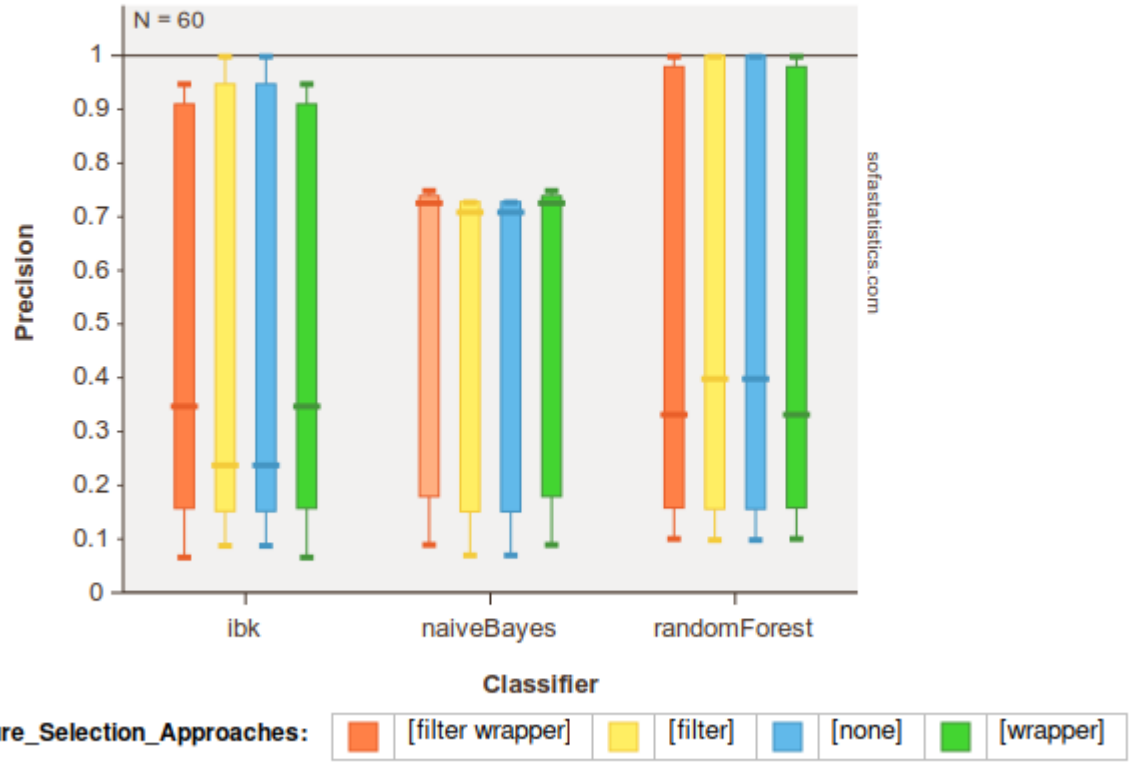
- Su Avro, per ibk e randomForest si ha un peggioramento della Precision a seguito dell'applicazione di Wrapper FS
- Per naiveBayes si ha invece un leggero miglioramento
- Solo su randomForest, è stata registrata un'eccezionale Precision a 1 tranne che per pochi outliers a 0.9 quando non è stato applicato Wrapper

Discussione Risultati - Precision

From sofa_db.all on 15/06/2022 at 05:50

Data filtered by: `Dataset_Name`='Torkin1_bookkeeper_50%_dataset' AND (`Is_Feature_Search_Backward?`='false' OR `Feature_Selection_Approaches`='[none]')

Precision by Classifier, series by FS approach



Outliers displayed. Lower whiskers are 1.5 times the Inter-Quartile Range below the lower quartile, or the minimum value, whichever is closest to the middle. Upper whiskers are calculated using the same approach.

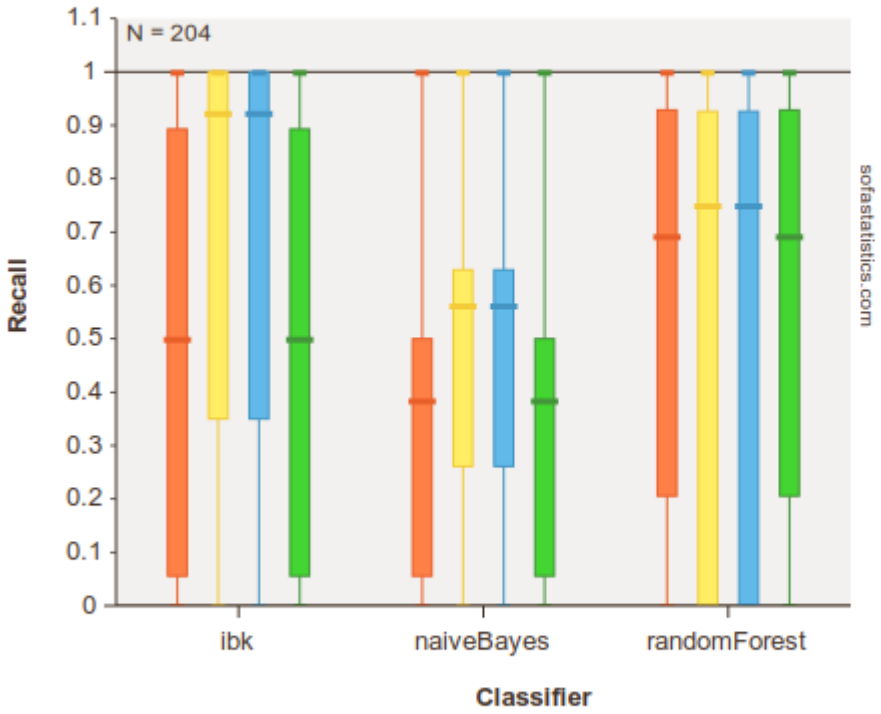
- In BookKeeper anche ibk mostra un valore medio della Precision più alto a seguito dell'applicazione di Wrapper, a fronte di un range di valori assunti leggermente ridotto

Discussione Risultati - Recall

From sofa_db.all on 15/06/2022 at 04:55

Data filtered by: `Dataset_Name`='Torkin1_avro_dataset_50%' AND `Is_Feature_Search_Backward?`='false'

Recall by Classifier, series by FS approach



Feature_Selection_Approaches: [filter wrapper] [filter] [none] [wrapper]

Outliers displayed. Lower whiskers are 1.5 times the Inter-Quartile Range below the lower quartile, or the minimum value, whichever is closest to the middle. Upper whiskers are calculated using the same approach.

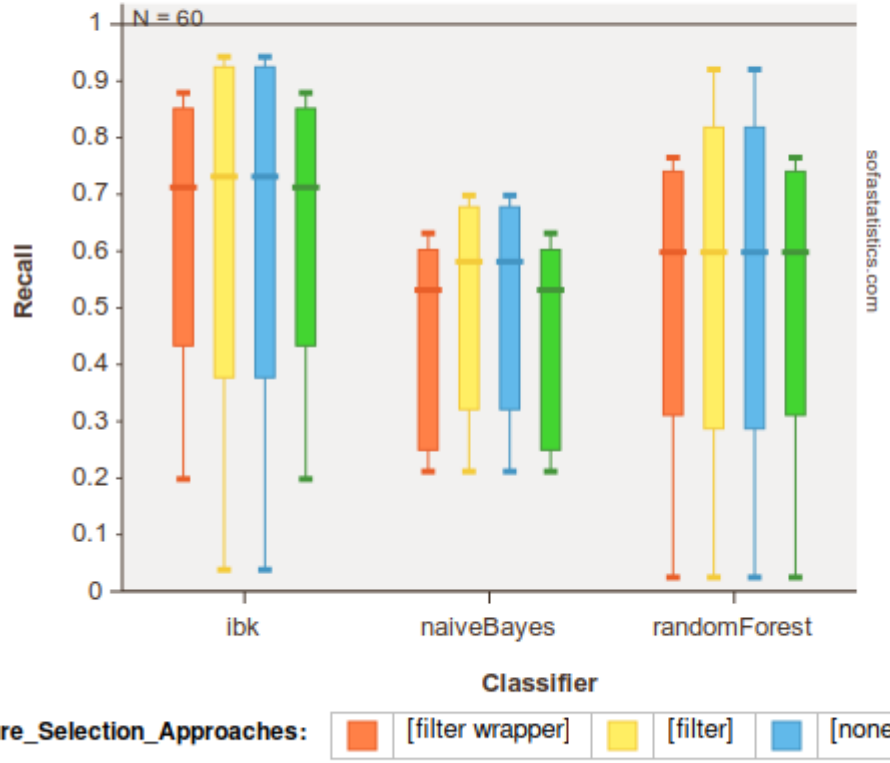
- Con avro, per quanto riguarda la Recall, si può assistere a un netto peggioramento del valore prestazionale per tutti e tre i classificatori
- Sembrerebbe che ridurre le features considerate abbia peggiorato la facoltà dei classificatori di distinguere i TPs

Discussione Risultati - Recall

From sofa_db.all on 15/06/2022 at 05:48

Data filtered by: `Dataset_Name`='Torkin1_bookkeeper_50%_dataset' AND (`Is_Feature_Search_Backward?`='false' OR `Feature_Selection_Approaches`='[none]')

Recall by Classifier, series by FS approach



Outliers displayed. Lower whiskers are 1.5 times the Inter-Quartile Range below the lower quartile, or the minimum value, whichever is closest to the middle. Upper whiskers are calculated using the same approach.

- Cin BookKeeper, si ha un leggero peggioramento delle prestazioni per il classificatore ibk, tuttavia il range dei valori assumibili della Recall è più ridotto, suggerendo una maggiore stabilità delle prestazioni
- Per naiveBayes, l'applicazione di Wrapper ha comportato un peggioramento della Recall
- Per randomForest, il valore medio della Recall assunto con tutte le combinazioni di Fs è lo stesso, tuttavia con l'approccio wrapper il range di valori assunti è più ristretto

Conclusioni

- Sembrerebbe che la feature PUBLICS **non sia poi così significativa** per la descrizione di un modello di dataset per la previsione di difetti nel codice, né presa singolarmente né in un sottoinsieme delle metriche considerate
  - È possibile che una classe con PUBLICS alto possa essere fonte di problemi non perché può ospitare al suo interno dei bug, piuttosto perché una sua modifica può essere causa di un bug regressivo da qualche altra parte
  - È possibile che esista qualche altra metrica qui non considerata che, congiuntamente a PUBLICS, possa creare un insieme significativo di features
  - Si sono rivelate features significative CODE_SMELLS, N_FIX, DEPENDENCIES
- Sembrerebbe che la FS sia **innocua** o perfino **controproducente** nella maggior parte dei casi considerati, per tutte le metriche prestazionali esaminate, poiché è difficile per i classificatori fare di meglio considerando un sottoinsieme degli attributi iniziali, in numero evidentemente già esiguo

Conclusioni – Approcci FS

- L’approccio Wrapper molto spesso porta un peggioramento delle metriche prestazionali, ed eccezionalmente un leggero miglioramento. Ciò è inaspettato, poiché tale approccio dovrebbe fornire il sottoinsieme di features più efficaci per il particolare classificatore su cui è usato
  - Le cause saranno approfondite in lavori futuri ...
- L’approccio Filter produce sempre le stesse prestazioni di quando non è applicata alcuna FS
  - Ciò indica che le features considerate oltre al sottoinsieme prodotto da FS non aiutano il classificatore, né lo ostacolano
- Combinare gli approcci Filter e Wrapper produce sempre le stesse prestazioni di quando è applicato solo Wrapper
  - Ciò indica che l’approccio Wrapper è quello che scarta via il maggior numero di features nei casi esaminati

Conclusioni – Classificatori

- NaiveBayes sembra il classificatore che ha avuto le prestazioni peggiori
  - Fanno eccezione la Precision, l'AUC e Kappa nel caso di BookKeeper. Tuttavia, si noti la ristrettezza di tale dataset
  - Ciò potrebbe significare che le features considerate siano molto correlate tra di loro
- Ibk senza FS è il classificatore che presenta la Recall media piu alta
- In generale, è difficile stabilire quale sia il miglior classificatore, poiché i valori delle metriche prestazionali sembrano variare molto in base al dataset considerato, e non c'è nessun classificatore che predomina in maniera decisiva sugli altri in tutte le metriche
- Nessuno dei classificatori è risultato mediamente peggiore rispetto a uno dummy, anche con FS wrapper

Grazie per l’attenzione!

Link ai sorgenti e analisi QA dei programmi utilizzati	`buggynessCrawler`	`classifier-evaluator`
GitHub	https://github.com/Torkin1/buggynessCrawler	https://github.com/Torkin1/classifier-evaluator/
SonarCloud	https://sonarcloud.io/summary/overall?id=Torkin1_buggynessCrawler	https://sonarcloud.io/summary/overall?id=Torkin1_classifier-evaluator